



(12) **EUROPEAN PATENT APPLICATION**

(43) Date of publication:
16.09.2015 Bulletin 2015/38

(51) Int Cl.:
G06F 9/455 (2006.01)

(21) Application number: **14167412.7**

(22) Date of filing: **04.08.2009**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO SE SI SK SM TR

- **Jeong, Bok-deuk**
449-712 Gyunggi-do (KR)
- **Suh, Sang-bum**
449-712 Gyunggi-do (KR)

(30) Priority: **06.08.2008 KR 20080076820**
22.05.2009 KR 20090044971

(74) Representative: **Land, Addick Adrianus Gosling Arnold & Siedsma**
Bezuidenhoutseweg 57
2594 AC Den Haag (NL)

(62) Document number(s) of the earlier application(s) in accordance with Art. 76 EPC:
09167145.3 / 2 154 610

Remarks:

This application was filed on 07-05-2014 as a divisional application to the application mentioned under INID code 62.

(71) Applicant: **Samsung Electronics Co., Ltd**
Gyeonggi-do 443-742 (KR)

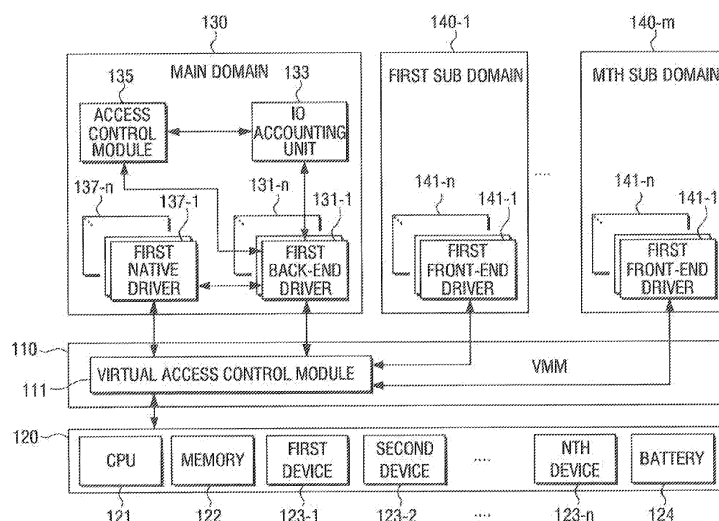
(72) Inventors:
• **Lee, Sung-min**
449-712 Gyunggi-do (KR)

(54) **Virtualization apparatus and method for controlling the same**

(57) A virtualization apparatus and a method for controlling the same. In a method for controlling a virtualization apparatus including a plurality of domains, a sub domain transmits an input/output (IO) request for a hard-

ware device to a main domain, and the main domain controls whether or not the IO request accesses the hardware device according to a resource needed to perform the IO request.

FIG. 1



Description

[0001] This application claims the benefit of Korean Application No. 2008-0076820, filed August 6, 2008, and No. 2009-0044971, filed May 22, 2009, in the Korean Intellectual Property Office, the disclosure of which is incorporated herein by reference.

[0002] One or more embodiments of the present invention relate to a virtualization apparatus and a method for controlling the same, and more particularly, to a virtualization apparatus that controls access of input/output (IO) requests, taking into consideration available resources for each device, and a method for controlling the same.

[0003] In computing, virtualization enables a plurality of operating systems to operate in a single computing environment. If it is assumed that an environment in which each operating system operates is a domain, each domain shares devices installed in the computing apparatus. However, when using the virtualization technology, a main domain among a plurality of domains has a native driver which can access the devices. Accordingly, sub domains may actually access the devices via the native driver of the main domain.

[0004] In order to implement such virtualization, a virtual machine monitor (VMM) initially allocates a central processing unit (CPU) time for each domain. Accordingly, if there is a request from a sub domain, the main domain uses a CPU time allocated to the main domain in order to process the request so that a CPU is occupied longer than the time initially allocated to the main domain.

[0005] In addition, if malware is installed in a sub domain, the malware sends excessive input/output (IO) requests and thereby the main domain consumes CPU resources while performing the job for the sub domain. That is, the main domain occupies the CPU due to a denial-of-service (DoS) attack from the malware on the sub domain, so performance of principal services or applications of the main domain deteriorates. Furthermore, excessive requests from the malware increase battery or power consumption.

SUMMARY

[0006] One or more embodiments of the present invention provide a virtualization apparatus which solves deterioration of performance of principal services provided in a virtualization environment, barrier to application performance, battery consumption and so on, and a method for controlling the same.

[0007] Additional aspects and/or advantages will be set forth in part in the description which follows and, in part, will be apparent from the description, or may be learned by practice of the invention.

[0008] According to an embodiment of the present invention, there is provided a method for controlling a virtualization apparatus including a main domain, one or more sub domains, and one or more hardware devices

The method includes receiving an input/output (IO) request for a hardware device from a sub domain, and controlling access to the hardware device by the IO request according to a resource needed to perform the IO request.

[0009] The controlling operation may include comparing the resource needed to perform the IO request with a reference value, and blocking the access to the hardware device if the resource needed to perform the IO request is greater than the reference value.

[0010] The controlling operation may include comparing the resource needed to perform the IO request with a reference value, and processing an amount of the IO request corresponding to the reference value if the resource needed to perform the IO request is greater than the reference value, and blocking the remaining amount of the IO request until the processing is over.

[0011] The resource needed to perform the IO request may be at least one of a size of the IO request, a central processing unit (CPU) time needed to perform the IO request and a maximum value of the CPU time needed to perform the IO request.

[0012] The method may further include checking a remaining capacity of a battery, and blocking the access to the hardware device if the remaining capacity of the battery is less than a reference value.

[0013] The method may further include detecting a malicious program code, and blocking the access to the hardware device by the IO request caused by the malicious program code.

[0014] According to another embodiment of the present invention, there is provided a virtualization apparatus, including a system resource unit comprising one or more hardware device, a main domain comprising a native driver for the hardware device, and one or more sub domain to transmit an input/output (IO) request for the hardware device to the main domain. The main domain may control access to the hardware device by the IO request according to a resource needed to perform the IO request.

[0015] The main domain may further include a back-end driver associated with the hardware device to receive the IO request from the sub domain.

[0016] The main domain may further include an access control module to compare the resource needed to perform the IO request with a reference value, and block access to the hardware device by the IO request if the resource needed to perform the IO request is greater than the reference value.

[0017] The main domain may further include an access control module to compare the resource needed to perform the IO request with a reference value, process an amount of the IO request corresponding to the reference value if the resource needed to perform the IO request is greater than the reference value, and block the remaining amount of the IO request until the processing is over.

[0018] The main domain may further include an IO accounting unit to calculate the resource needed to perform

the IO request.

[0019] The resource needed to perform the IO request may be at least one of a size of the IO request and a central processing unit (CPU) time needed to perform the IO request.

[0020] The main domain may further include an access control module to check a remaining capacity of a battery and block the access to the hardware device if the remaining capacity of the battery less than a reference value.

[0021] The main domain may further include an access control module to detect a malicious program code and block the access to the hardware device by the IO request caused by the malicious program code.

[0022] The virtualization apparatus may be a consumer electronic device.

BRIEF DESCRIPTION OF THE DRAWINGS

[0023] These and/or other aspects and advantages of the invention will become apparent and more readily appreciated from the following description of the embodiments, taken in conjunction with the accompanying drawings of which:

FIG. 1 is a schematic block diagram of a virtualization apparatus, according to an embodiment of the present invention;

FIG. 2 is a flow chart illustrating initialization for access control in a main domain, according to an embodiment of the present invention; and

FIG. 3 is a flow chart illustrating an access control method of a virtualization apparatus, according to an embodiment of the present invention.

DETAILED DESCRIPTION

[0024] Reference will now be made in detail to embodiments of the present invention, examples of which are illustrated in the accompanying drawings, wherein like reference numerals refer to the like elements throughout. Embodiments are described below in order to explain the present invention by referring to the figures.

[0025] FIG. 1 is a schematic block diagram of a virtualization apparatus, according to an embodiment of the present invention.

[0026] Referring to FIG. 1, the virtualization apparatus includes a virtual machine monitor (VMM) 110, a system resource unit 120, a main domain 130, and a plurality of sub domains 140-1, ..., and 140-m.

[0027] Physical embodiments of virtualization apparatus can include for example, consumer electronic (CE) devices such as personal computers (e.g. desktop computers or laptop computers), audio equipment and mobile devices (e.g. cell phones or smart phones.)

[0028] The VMM 110 may execute a plurality of operating systems concurrently. In a virtualization environment using the VMM 110, back-end drivers 131-1, ...,

and 131-n and native drivers 137-1, ..., and 137-n are located in the main domain 130, and front-end drivers 141-1, ..., and 141-n are located in respective sub domains 140-1, ..., and 140-m. A virtual access control module 111 in the VMM 110 limits CPU usage for each domain 130, 140-1, ..., and 140-m.

[0029] The system resource unit 120 constitutes hardware of the virtualization apparatus, and may include a central processing unit (CPU) 121, a memory 122, a first to nth device 123-1, ..., and 123-n, and a battery 124. It may further include a plurality of other devices and/or storages.

[0030] The CPU 121 processes operations in a device upon a virtualization apparatus.

[0031] The memory 122 may store a resource estimation formula for calculating a CPU usage needed to perform an input/output (IO) request, that is, a resource for each device 123-1, ..., and 123-n or for a plurality of IO requests for each device 123-1, ..., and 123-n. CPU usage refers to a resource to perform an IO request, such as a CPU usage time or a CPU time.

[0032] In addition, in the memory 122, an access control policy needed to control accesses to devices corresponding to IO requests output from the plurality of sub domains 140-1, ..., and 140-m are loaded and stored. The access control policy includes a reference value to control the mth sub domain 140-m to access the nth device 123-n according to an IO request transmitted from the mth sub domain 140-m, or to schedule an access method thereof.

[0033] The access control policy may set a threshold value for IO request size or CPU usage available for each device 123-1, ..., and 123-n installed in the system resource unit 120. In other words, the threshold value may be a maximum value (or a reference value) of the IO request size or the CPU usage below which the access control policy allows access to each device 123-1, ..., and 123-n. For example, the access control policy may set a maximum value (or a reference value) of the available CPU usage for a predetermined period of time for an IO request by each front-end driver 141-1, ..., and 141-n, wherein the predetermined period of time may represent a CPU scheduling period. Hereinbelow, a reference set for each device 123-1, ..., and 123-n is described as an example.

[0034] The first to nth devices 123-1, ..., and 123-n are IO devices that perform a function for inputting information to a device, a function for outputting information from a device, or a function for performing input and output, including for example network interface cards, storages, memory technology devices (MTDs), and console devices.

[0035] The battery 124 is included to supply power to the system resource unit 120.

[0036] The main domain 130 and the one or more sub domains 140-1, ..., and 140-m provide virtualization environments which are operated by separate operating systems. The main domain 130 and the one or more sub

domains 140-1, ..., and 140-m may also operate in the same operating system. Examples of operating systems include Palm, Symbian OS, Android, Windows Mobile, Windows(R), Linux, Unix(TM), and the disk operating system (DOS).

[0037] The main domain 130 provides principal services such as Internet banking, security services, and voice calls, or main applications. The main domain 130 provides the plurality of sub domains 140-1, ..., and 140-m as well as the main domain 130 with paths to access the first to nth devices 123-1, ..., and 123-n. Accordingly, the main domain 130 functions as a server, and the plurality of sub domains 140-1, ..., and 140-m function as clients.

[0038] To this end, the main domain 130 may include first to nth back-end drivers 131-1, ..., and 131-n, an IO accounting unit 133, an access control module 135, and first to nth native drivers 137-1, ..., and 137-n.

[0039] The first to nth back-end drivers 131-1, ..., and 131-n each provide a communication path between the main domain 130 and the plurality of sub domains 140-1, ..., and 140-m, and correspond respectively to the first to nth devices 123-1, ..., and 123-n. The first to nth back-end drivers 131-1, ..., and 131-n each provide the IO accounting unit 133 with IO requests input from the plurality of sub domains 140-1, ..., and 140-m.

[0040] For example, if an application (not shown) on the first sub domain 141-1 requests the use of the first device 123-1, the first front-end driver 141-1 transmits an IO request for the first device 123-1, and the first back-end driver 131-1 receives the IO request and transmits the IO request to the IO accounting unit 133. The first device 123-1 is a target device.

[0041] IO accounting unit 133 provides a formula to estimate how much CPU resources are used for the IO request. If the memory 122 does not have a formula for a newly installed device or if there is a request for generating a new formula, the IO accounting unit 133 generates a resource estimation formula. For example, the IO accounting unit 133 newly generates a resource estimation formula: when an apparatus to which a virtualization environment is applied is manufactured; when a new device is installed in the system resource unit 120; or when the user requests an update of the existing formula.

[0042] If there is a request for generating a formula, the IO accounting unit 133 calculates a formula for estimating how much resources each device 123-1, ..., and 123-n uses for the IO request. A method for calculating the formula is as follows.

[0043] If mth sub domain 140-m transmits IO requests of diverse sizes, which are predetermined for each device 123-1, ..., and 123-n, to the main domain 130, the IO accounting unit 133, based on resources, for example a CPU time, needed to process the IO requests of diverse sizes, outputs a resource estimation formula of mth sub domain 140-m for each device 123-1, ..., and 123-n or each front-end drivers 141-1, ..., and 141-n, .

[0044] That is, if the nth front-end driver 141-n transmits test packets of diverse sizes to the nth back-end driver 131-n, the IO accounting unit 133 tests the amount of variance of a CPU time taken for the nth native driver 137-n or the nth device 123-n to process each packet, so a regression formula, that is, a resource estimation formula is output.

[0045] The resource estimation formulas for the devices 123-1, ..., and 123-n are set in the access control module 135 or memory 122, and are used for device access control when driving each device 123-1, ..., and 123-n.

[0046] The access control module 135 controls IO requests received from the sub domains 140-1, ..., and 140-m to access the main domain 130 or the devices 123-1, ..., and 123-n using the resource estimation formulas for the devices 123-1, ..., and 123-n. That is, the access control module 135 may control access of the IO request or schedule an access by comparing a resource needed to perform the IO request calculated by a corresponding resource estimation formula with a stored reference value.

[0047] In particular, if an IO request for the nth device 123-n is received from the mth sub domain 140-m, and a resource needed to perform the IO request is greater than a reference value for the nth device 123-n stored in the memory 122, the access control module 135 may process an amount of the IO request corresponding to the reference value, then blocks the remaining amount of the IO request, and stores the remaining amount of the IO request in an additional storage device such as a queue (not shown).

[0048] For example, if it is assumed that a resource needed to process a packet associated with an IO request for the first device 123-1 is 10 msec, and a reference value for the first device 123-1 is 2 msec, the access control module 135 firstly processes the amount of the IO request corresponding to 2 msec, and blocks the remaining amount of the IO request corresponding to 8 msec while processing the amount of the IO request corresponding to 2 msec. After finishing processing the amount of the IO request corresponding to 2 msec, the access control module 135 processes 2 msec from among the remaining 8 msec, and blocks the remaining 6 msec. The remaining 6 msec is processed in the same manner.

[0049] The access control module 135 may use the IO request size as the resource needed to perform the IO request. For example, if it is assumed that a packet size associated with an IO request for the first device 123-1 is 4MB(mega byte), and a reference value for the first device 123-1 is 1MB, the access control module 135 firstly processes the amount of the IO request corresponding to 1MB, and blocks the remaining amount of the IO request corresponding to 3MB while processing the amount of the IO request corresponding to 1MB. After finishing processing the amount of the IO request corresponding to 1MB, the access control module 135 processes 1MB from among the remaining 3MB, and blocks

the remaining 2MB. The remaining 2MB is processed in the same manner.

[0050] Alternatively, if a resource needed to perform an IO request is k times as much as a reference value stored in the memory 122, the access control module 135 may determine that the IO request is an attack by malware and intercept the IO request so that the IO request cannot be processed. The value 'k' may be set or modified by a system designer or the user. Malware may be installed in at least one of the sub domains 140-1, ..., and 140-m.

[0051] The first to nth native drivers 137-1, ..., and 137-n are actual drivers accessible to the first to nth devices 123-1, ..., and 123-n installed in the system resource unit 120. The first to nth native drivers 137-1, ..., and 137-n correspond to the first to nth devices 123-1, ..., and 123-n, respectively.

[0052] The first to mth sub domains 140-1, ..., and 140-m (m is a constant) may access the main domain 130 and the first to nth devices 123-1, ..., and 123-n via the first to nth front-end drivers 141-1, ..., and 141-n. The first to nth back-end drivers 131-1, ..., and 131-n and the first to nth front-end drivers 141-1, ..., and 141-n are associated with the first to nth native drivers 137-1, ..., and 137-n and the first to nth devices 123-1, ..., and 123-n, and may be automatically generated when the first to nth native drivers 137-1, ..., and 137-n of the first to nth devices 123-1, ..., and 123-n are installed.

[0053] FIG. 2 is a flow chart illustrating initialization for access control in a main domain, according to an embodiment of the present invention.

[0054] Referring to FIGS. 1 and 2, if there is a request for generating a resource estimation formula for the nth device 123-n (S210-Y), the nth front-end driver 141-n transmits IO requests of diverse sizes which are defined for the nth device 123-n to the nth back-end driver 131-n of the main domain 130, and the nth back-end driver 131-n provides the IO accounting unit 133 with the IO requests of diverse sizes (S220).

[0055] Subsequently, the IO accounting unit 133 generates a resource estimation formula for the nth device 123-n using resources taken to process the IO requests of diverse sizes (S230). In addition, the IO accounting unit 133 calculates the sum of data (for example, CPU time) which a certain domain requests to use a certain device for every CPU scheduling period. Operations S210 to S230 are performed to generate resource estimation formulas for the first to nth devices 123-1, ..., and 123-n, but for convenience of description, only the nth device 123-n is recited here.

[0056] If the resource estimation formulas for the first to nth devices 123-1, ..., and 123-n are generated, the access control module 135 sets the generated resource estimation formulas separately for each device 123-1, ..., and 123-n in the access control module 135 or the memory 122 (S240).

[0057] The access control module 135 loads a access control policy in the memory 122 (S250).

[0058] In operation S210, if there is no request for generating a resource estimation formula for the nth device 123-n (S210-N), the access control module 135 determines whether or not an access control policy is loaded in the memory 122 (S260). If an access control policy is not loaded in the memory 122 (S260-N), the access control module 135 loads an access control policy in the memory 122. By this process, initialization for access control in response to an IO request is completed.

[0059] Such a virtualization apparatus and a method for controlling the same according to one or more embodiments of the present invention can solve performance deterioration of principal services or applications, and minimize unnecessary battery consumption.

[0060] FIG. 3 is a flow chart illustrating an access control method of a virtualization apparatus, according to an embodiment of the present invention.

[0061] Referring to FIGS. 1 to 3, if an IO request is transmitted from one of the front end drivers 141-1, ..., and 141-n of the first to mth sub domains 140-1, ..., and 140-m to the main domain 130 (S310), the IO accounting unit 133 calculates resources used for the IO request using a resource estimation formula corresponding to the IO request (S320).

[0062] For example, if the nth front-end driver 141-n transmits an IO request, the nth back-end driver 131-n receives the IO request and transmits the IO request to the IO accounting unit 133. The IO accounting unit 133 reads out a resource estimation formula corresponding to the nth device 123-n from the memory 122, and puts the IO request into the read-out resource estimation formula, so a resource used for the IO request is calculated. In addition, the IO accounting unit 133 calculates the sum of data (CPU time) which a certain domain requests to use a certain device for every CPU scheduling period. The sum of the data is reset to be 0 in the next CPU scheduling period.

[0063] The access control module 135 compares the resources calculated in operation S320 with a reference value of an access control policy corresponding to the nth device 123-n, which is loaded in the memory 122, and thereby determines whether or not the nth device 123-n can perform the IO request (S330). The access control policy may specify how much CPU time the main domain 130 can use to process the IO request for every CPU scheduling period so that a certain sub domain can use a certain device provided by the main domain 130.

[0064] If the calculated resource is smaller than the reference value corresponding to the nth device 123-n (S330-Y), the access control module 135 determines that the nth device 123-n can perform the IO request, and thus the IO request can be performed (S340). For example, the access control module 135 transmits the IO request to the nth device 123-n through the nth back-end driver 131-n and the nth native driver 137-n so that an operation corresponding to the IO request can be performed.

[0065] Accordingly, operation S340 is performed when

the sum of data (CPU time) which a certain domain requests to use a certain device for every CPU scheduling period is smaller than the reference value specified in the access control policy.

[0066] Alternatively, if the calculated resource is higher than the reference value corresponding to the nth device 123-n (S330-N), the access control module 135 limits the use of the resource according to the access control policy (S350). In particular, the access control module 135 may sequentially process an amount of the IO request up to the reference value allowed in the access control policy for every scheduling period, or block the IO request for a predetermined period of time. For example, the access control module 135 may sequentially perform the IO request up to the IO request size or the CPU time of the access control policy, which is loaded in the memory 122. In addition, if it is determined that the IO request is a malware attack, the access control module 135 intercepts access of the IO request so that access to the hardware device by the IO request (for example, the nth device 123-n) can be controlled.

[0067] In embodiments of the present invention, the access control module 135 or a sensor (not shown) may check the remaining life of a battery, and the access control policy may include an access control policy commensurate with the remaining life of the battery. If the checked remaining battery life reaches the remaining battery life threshold set in the access control policy, the access control module 135 may limit the CPU usage. In the access control policy, the different remaining battery life for each device 123-1, ..., and 123-n may be set. The access control module 135 may limit the CPU usage for each device 123-1, ..., and 123-n taking into consideration the different remaining battery life.

[0068] Furthermore, the main domain 130 may include an invasion detection module (not shown) which dynamically detects invasion of malicious program code such as malware. If the invasion detection module detects invasion of malicious program code, the access control module 135 may not allow access of the IO request.

[0069] As described above, according to embodiments of the present invention, available resources for each device 123-1, ..., and 123-n are preset, and access to the main domain 130 or access to each device 123-1, ..., and 123-n is controlled according to the IO request, so fine-grained IO access control (FGAC) may be provided. That is, in order to provide the FGAC, embodiments of the present invention can limit use of resources such as a CPU based on the access control policy predetermined for each device 123-1, ..., and 123-n when the main domain 130 processes the IO request from the mth sub domain 140-m.

[0070] Moreover, deterioration of system performance and unnecessary battery and CPU consumption caused due to excessive IO requests of malware installed in the virtualization apparatus may be prevented. That is, access to devices can be restricted or blocked according to IO requests so that unnecessary waste of resources

can be prevented.

[0071] Furthermore, as access control divided according to each device 123-1, ..., and 123-n is provided, there are no further resource restrictions placed on devices that do not receive denial-of-service (DoS) attacks. In addition, principal applications can operate using optimum resources.

[0072] In particular, if the virtualization apparatus is a consumer electronics device having limited resources such as a cell phone, a DoS attack may prove fatal. Accordingly, the virtualization apparatus can be protected from DoS attacks by supporting FGAC, that is, access control for each device.

[0073] Although a few embodiments of the present invention have been shown and described, it would be appreciated by those skilled in the art that changes may be made in these embodiments without departing from the principles and spirit of the invention, the scope of which is defined in the claims and their equivalents.

Claims

1. A virtualization apparatus, comprising:

a system resource unit comprising one or more hardware device;
a main domain comprising a native driver for the hardware device; and
one or more sub domain to transmit an input/output (IO) request for the hardware device to the main domain,
wherein the main domain controls access to the hardware device by the IO request according to a resource needed to perform the IO request.

2. A virtualization apparatus according to claim 1, wherein the main domain further comprises an access control module (135) adapted to compare the resource needed to perform the IO request with a reference value, and to block access to the hardware device by the IO request if the resource needed to perform the IO request is greater than the reference value,
wherein the resource needed to perform the IO request is at least a size of the IO request and/or CPU-time (central processing unit) needed to perform the IO request.
3. A virtualization apparatus according to claim 2, wherein the main domain further comprises an IO accounting unit (133) adapted to calculate the resource needed to perform the IO request.
4. The virtualization apparatus according to claim 1, 2 or 3, wherein the main domain further comprises a back-end driver (131-1, 131-n) associated with the hardware device adapted to receive the IO request

from the sub domain.

5. The virtualization apparatus according to any of claims 1-4, wherein the main domain further comprises an access control module adapted to check a remaining capacity of a battery (124) and to block the access to the hardware device if the remaining capacity of the battery less than a reference value. 5
6. The virtualization apparatus according to any of claims 1-5, wherein the main domain further comprises an access control module adapted to detect a malicious program code and to block the access to the hardware device by the IO request caused by the malicious program code. 10
7. The visualization apparatus according to any of claims 1-6, wherein the virtualization apparatus is a consumer electronic device. 20
8. A method for controlling a virtualization apparatus comprising a main domain, one or more sub domains, and one or more hardware devices, the method comprising: 25
 - receiving an input/output(IO) request for a hardware device from a sub domain; and
 - controlling access to the hardware device by the IO request according to a resource needed to perform the IO request. 30
9. The method according to claim 8, wherein the controlling operation comprises: 35
 - comparing the resource needed to perform the IO request with a reference value; and
 - blocking the access to the hardware device if the resource needed to perform the IO request is greater than the reference value. 40
10. The method according to claim 8 or 9, wherein in the controlling operation comprises: 45
 - comparing the resource needed to perform the IO request with a reference value; and
 - processing an amount of the IO request corresponding to the reference value if the resource needed to perform the IO request is greater than the reference value, and blocking the remaining amount of the IO request until the processing is over. 50
11. The method for controlling a virtualization apparatus according to any of the claims 1-7, wherein the resource needed to perform the IO request is at least one of a size of the IO request, a central processing unit (CPU) time needed to perform the IO request and a maximum value of the 55

CPU time needed to perform the IO request.

12. The method according to claim 10, further comprising:
 - checking a remaining capacity of a battery (124); and
 - blocking the access to the hardware device if the remaining capacity of the battery is less than a reference value.
13. The method according to any of claims 8-12, further comprising:
 - detecting a malicious program code; and
 - blocking the access to the hardware device by the IO request caused by the malicious program code.

FIG. 1

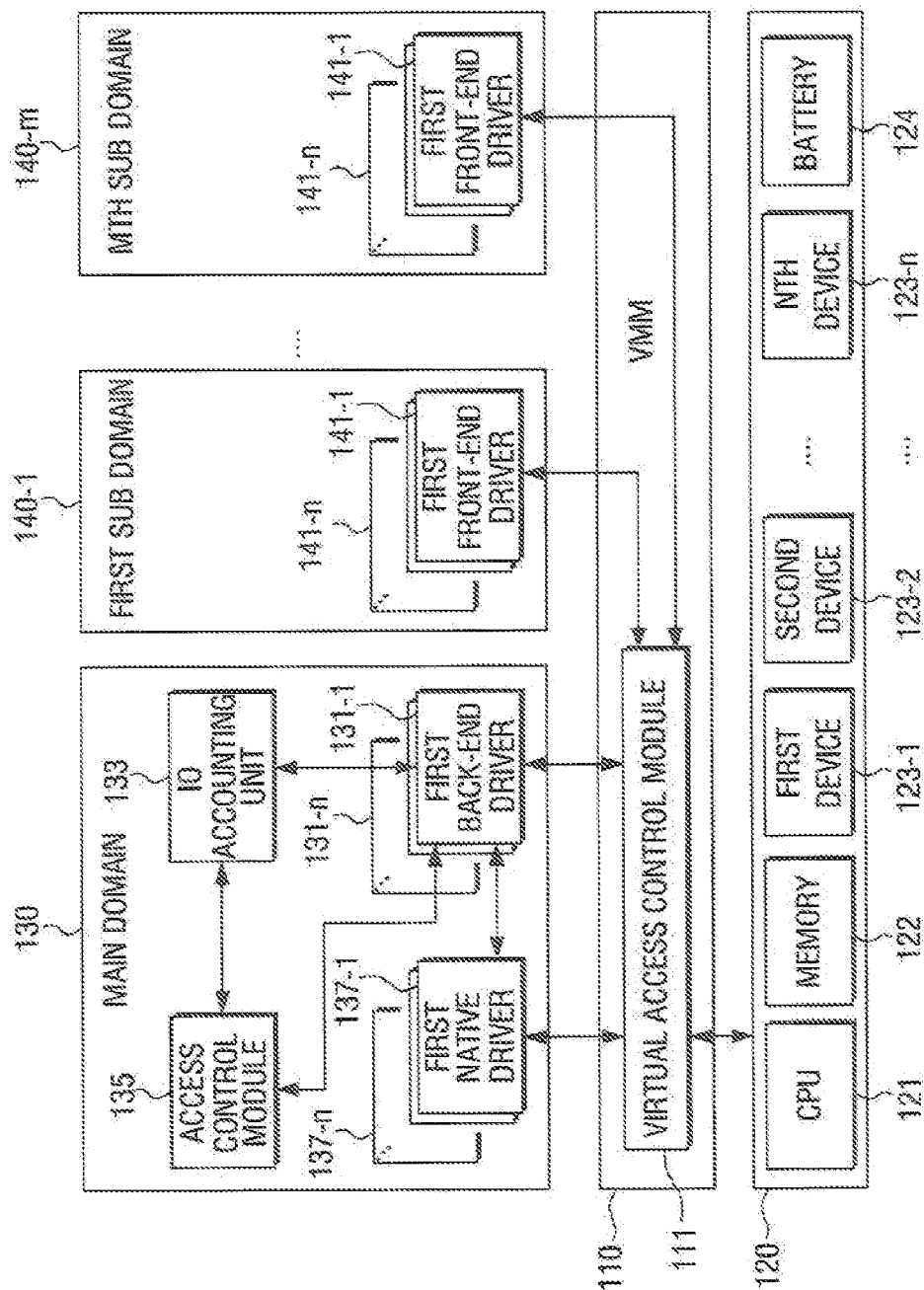


FIG. 2

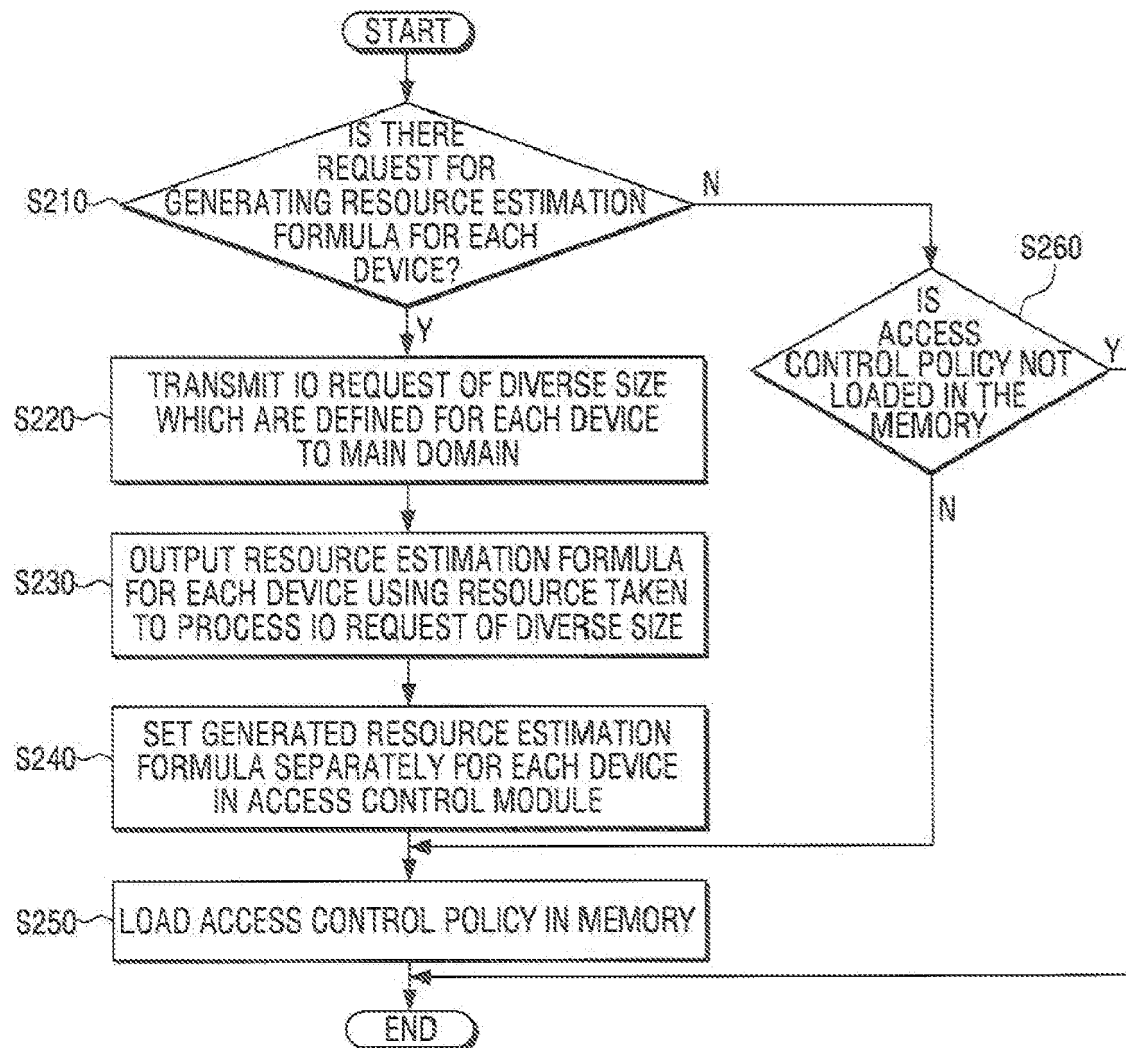
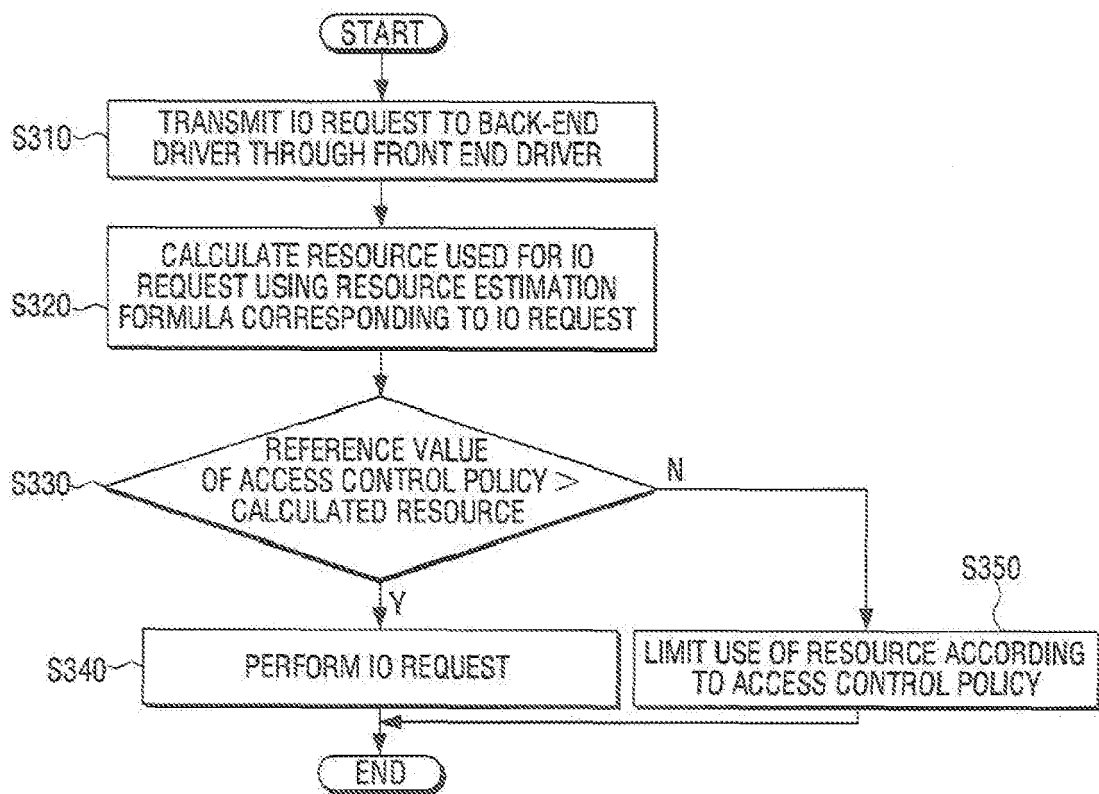


FIG. 3





EUROPEAN SEARCH REPORT

Application Number
EP 14 16 7412

5

10

15

20

25

30

35

40

45

50

55

DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	Sang-Bum Suh: "Secure Architecture and Implementation of Xen on ARM for Mobile Devices", 17 April 2007 (2007-04-17), pages 1-48, XP055088508, Xen Summit Spring 2007, IBM TJ Watson Retrieved from the Internet: URL: http://xen.org/files/xensummit_4/Secure_Xen_ARM_xen-summit-04_07_Suh.pdf [retrieved on 2013-11-15] * the whole document * * in particular pages 5-6, 10, 24 and 27-28 * -----	1-13	INV. G06F9/455
			TECHNICAL FIELDS SEARCHED (IPC) G06F
The present search report has been drawn up for all claims			
Place of search		Date of completion of the search	Examiner
Munich		27 May 2015	Steinmetz, Christof
CATEGORY OF CITED DOCUMENTS X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure P : intermediate document T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons & : member of the same patent family, corresponding document			

EPO FORM 1503 03.82 (P04C01)

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- KR 20080076820 [0001]
- KR 20090044971 [0001]